PHYSICAL MODELING MEETS MACHINE LEARNING: TEACHING BOW CONTROL TO A VIRTUAL VIOLINIST

Graham Percival
Science and Music Research Group
University of Glasgow, UK
graham@percival-music.ca

Nicholas Bailey
Science and Music Research Group
University of Glasgow, UK
nick@n-ism.org

George Tzanetakis
Department of Computer Science
University of Victoria, Canada
gtzan@cs.uvic.ca

ABSTRACT

The control of musical instrument physical models is difficult; it takes many years for professional musicians to learn their craft. We perform intelligent control of a violin physical model by analyzing the audio output and adjusting the physical inputs to the system using trained Support Vector Machines (SVM).

Vivi, the virtual violinist is a computer program which can perform music notation with the same skill as a beginning violin student. After only four hours of interactive training, *Vivi* can play all of Suzuki violin volume 1 with quality that is comparable to a human student.

Although physical constants are used to generate audio with the model, the control loop takes a "black-box" approach to the system. The controller generates the finger position, bow-bridge distance, bow velocity, and bow force without knowing those physical constants. This method can therefore be used with other bowed-string physical models and even musical robots.

1. INTRODUCTION

One well-known problem with physical modelling of musical instruments is control [1] – a musical instrument is a non-linear dynamical system; some physical models even include non-deterministic elements. Producing good sound with such a system is difficult; professional classical musicians spend more than ten years learning their instruments before they can earn a living by performing. Therefore, we can expect that autonomous control of a physical model will require a great deal of research and training.

Rather than trying to reproduce the amount of control exhibited by professional musicians, we created a computer program which can be trained in a manner similar to a human violin student: by receiving feedback from a teacher and "listening" to its output. After four hours of interactive training, *Vivi, the virtual violinist* can play at approximately the level of a student with one year of experience.

This is achieved by asking a human user to classify a series of audio examples created using a violin physical model. The human judgements are used to train Support Vector Machine (SVM) classifiers, which are used to adjust the bowing parameters when performing music (Figure 1). After a short period of "basic training", *Vivi* can practise scales and pieces of music. For each performance, the human may identify any part of the audio which needs adjustment; these corrections are used to re-train the SVM, which can then perform the music again. Such corrections will influence every subsequent performance; any training on simple scales will apply to a Bach partita.

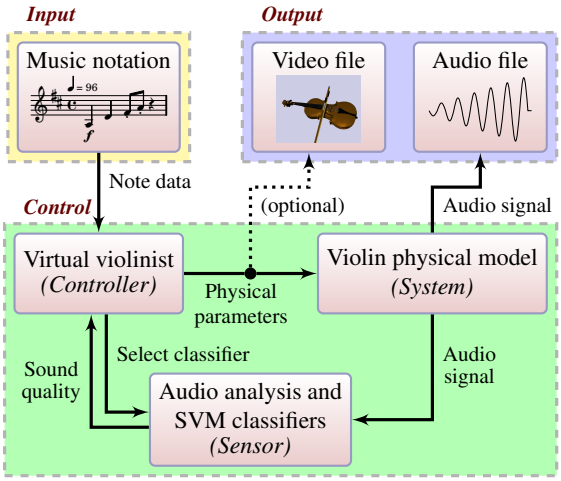

Figure 1. Music performance with *Vivi*.

1.1 Motivation

At the moment, producing a decent violin sound can only be done by highly-trained musicians. Playing a violin with good intonation and good sound quality requires very accurate fine muscle control. This level of muscle control may be taken for granted by professional musicians, but it is a serious problem for skilled amateurs and students; many people who studied classical music as children stop playing music later in life. In addition, violin playing can become a physical impossibility due to advanced age or various medical conditions.

Our goal is to enable anybody to create violin music, regardless of physical disability or lack of training – the only barrier to producing music should be the desire to do so. Somebody who would be satisfied with a "generic" performance should only be required to supply the sheet music; somebody wishing to customized a performance should only be required to specify high-level expressive gestures.

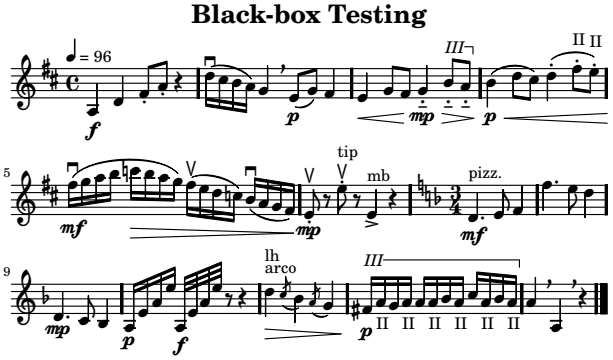

Figure 2. Music notation understood by *Vivi*.

Our work is also motivated by Miku Hatsune and the Vocaloid singing synthesis [2]. By eliminating the physical constraints of singing in a particular vocal range and style (i.e. Japanese pop music), users of this software can create vocal music which was previously impossible for them to produce. This has resulted in an extremely impressive range of music and videos from online collaborations [3].

1.2 Pedagogical inspiration and design goals

As the first author learned cello with the Suzuki method, our design draws upon some of that philosophy:

- Students do not practise alone; a parent should be involved in the daily practice.
- Students learn music from Suzuki violin book 1 [4], with a corresponding emphasis on Baroque music.
- The left-hand fingering is given in the sheet music; although Suzuki students do not read sheet music at the beginning, parents ensure that the student follows the printed fingering.
- The bow-bridge distances are set according to the dynamic. These are sometimes referred to as "bow lanes", or the "Kreisler highway" [5].
- The amount of bow for each note is often specified by the teacher (such as "half" or "quarter bow"), and the tempo is given by the piano accompaniment.
- Students do not know physical values such as the characteristic impedance Z_c of the string.
- Students are not expected to give expressive music performances; a "robotic" performance is an acceptable place to start.

We do not encourage inexpressive music, but a virtual violinist can be useful even without expressive performances. We again consider Vocaloid to be an inspiration: musicality can be added by the user while the software itself strictly follows the given instructions.

To improve our testing framework, we wrote a piece of music (Figure 2) which contains all the supported notation.

1.3 Sound and video examples

Examples of sheet music performed by *Vivi* are available [1]. In addition to music, the website contains audio examples which illustrate technical aspects of this paper.

[1] http://percival-music.ca/smc2011.html

Section 2 gives an overview of related work. The rest of the paper is generally structured from the fastest units of time to the slowest. Physical modeling (which generates samples at 44100 Hz) is described in Section 3. The control cycle (which modifies physical actions at 172 Hz) is covered in Section 4. Training (which does not occur at a constant rate) is covered in Section 5, while Section 6 covers performance rules (dealing with score events occurring at approximately 1-4 Hz). We end with a few implementation details in Section 7 and the conclusion in Section 8.

2. RELATED WORK

Intelligent control of physical bowing parameters would be impossible without a synthesis engine. We chose to use the bowed-string physical modelling algorithm with modal synthesis as described in Demoucron's Ph.D. thesis [6]. This algorithm does not include some of the more advanced knowledge of bow-string friction interaction [7], the effects of torsional waves in a bowed string [8], or the plastic thermal model of rosin friction [9], but it sounds sufficiently "violin-like" for our needs. The main alternative to modal synthesis is digital waveguide synthesis [7].

An alternate method of creating violin sound, without using a physical model, is concatenative synthesis (or Spectral Modeling Synthesis). This is an active research topic, with recent dissertations on predictive spectral envelopes for violin sounds [10], reconstructing violin bowing with Bézier curves [11], and further research combining these projects to synthesize violin sound [12].

There has been considerable interest in the bowing actions required to establish and maintain a good violin tone. An early examination of the bow force required to reach Helmholtz motion gave rise to "Schelleng diagrams" [13], while an examination of the initial attacks produced "Guettler diagrams" [14]. These have been re-examined with more accurate measurements from real musicians and bowing machines [15]. Some teachers are applying this scientific knowledge to violin pedagogy [5].

Other work has focused on the control of musical instruments. Proportional-integral-derivative (PID) controllers have been used to control a plucked string simulation and acoustic instruments [16, 17]. The famous flute-playing robot WF-4RIV [18] uses an artificial neural network to generate expressive music, and uses a control loop based on the relative strength of even- and odd-harmonics, and the overall sound intensity. The WF-4RIV also remembers its own mistakes while performing a piece, and adjusts future performances of that piece of music accordingly. Another group worked on a robotic violin bowing arm [19]; this project focused on reproducing the same sound when bowing the open D string multiple times, and analyzed the resulting sound by visually comparing the amplitude plots and spectograms.

Finally, objective analysis of violin tone quality has been performed, allowing computers to classify sounds with high accuracy. One project focused on judging sound quality of violin instruments [20], while another classified a *legato* violin note as being performed by a professional musician or a student, and recognized mistakes in bow control [21].

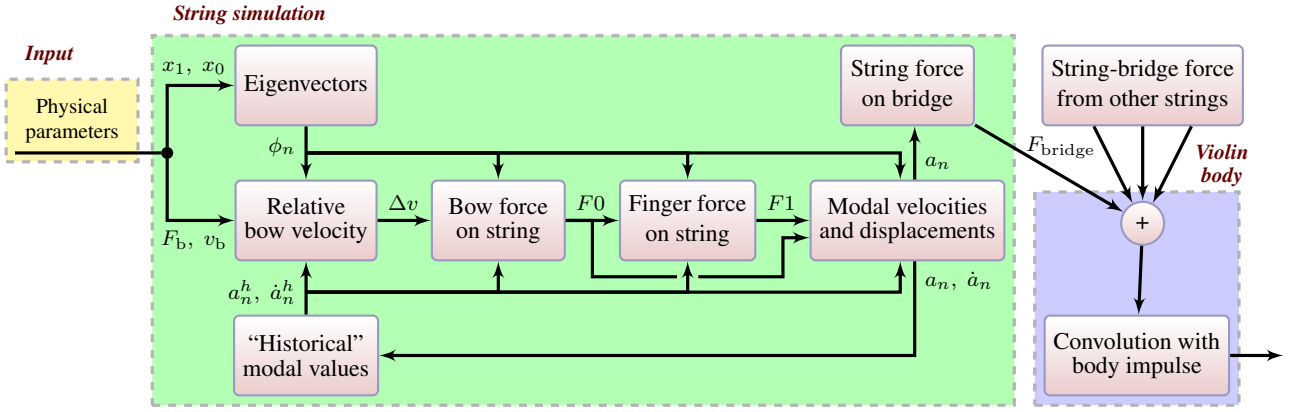

Figure 3. Overview of the violin physical modelling. Variables with a subscript$_n$ indicate an N-element vector, lines without any text indicate samples of an audio signal, and all other variables are individual numbers.

3. VIOLIN PHYSICAL MODELING

We wrote Artifastring ("artificial fast string"), which implements the bowed-string physical modelling with modal synthesis as described in [6]. Artifastring was written in C++, and is freely available [2] under the GPLv3. We hope that it can be of use to other researchers without detailed knowledge of acoustics, so that more people can work on bowed string performance. To support widespread usage, SWIG bindings [3] are also available.

An overview of the physical model is given in Figure 3, and a summary of the variables used is given in Table 1. We will cover only the most important equations here; for the full details, see [6] or the Artifastring source code.

The synthesis begins with a stiff string with linear density ρ_{L}, tension T, Young's modulus E, diameter d, and second moment of area for a circular cross-section $I = \frac{\pi d^4}{64}$. With the sum of external forces $F(x,t)$, the wave equation is:

$$\rho_{\text{L}}\frac{\partial^2 y(x,t)}{\partial t^2} - T\frac{\partial^2 y(x,t)}{\partial x^2} + EI\frac{\partial^4 y(x,t)}{\partial x^4} = F(x,t) \quad (1)$$

This gives the dispersion relation with string length L:

$$\omega_{0n} = \sqrt{\frac{T}{\rho_{\text{L}}}\left(\frac{n\pi}{L}\right)^2 + \frac{EI}{\rho_{\text{L}}}\left(\frac{n\pi}{L}\right)^4} \quad (2)$$

We define the modal displacement $a_n(t)$ and force $f_n(t)$ in terms of $\phi_n(x)$. We then add a damping coefficient $r_n = B_1 + B_2(n-1)^2$ to simulate various losses along the string. Solving (1) and (2) gives us an an infinity of equations, $n = 1 \ldots \infty$. We limit n to 100 as a compromise between sound quality and computational efficency.

$$\ddot{a}_n(t) + 2r_n\dot{a}_n(t) + w_{0n}^2 a_n(t) = \rho_{\text{L}}^{-1} f_n(t) \quad (3)$$

The calculation of each time step begins with $a_n^h(t)$ and $\dot{a}_n^h(t)$; these are the new modal values if no external forces are applied. To maintain a consistent terminology with [6], we use his term "historical" and the ^{h} superscript, but we believe that "free oscillation" would be more clear.

[2] http://percival-music.ca/artifastring/
[3] http://www.swig.org/

The external force $F0$ on the string therefore represents the extra force that must be applied to make the string at point x_1 move as described by the relative bow velocity Δv. If the bow is sticking to the string, $\Delta v = 0$. If the bow is slipping, a hyperbolic friction curve is used with the coefficients of static friction μ_s, dynamic friction μ_d, slope of the hyperbolic curve v_0. Noise was added by picking a random value $0.95 \leq N(t) \leq 1.0$ for each calculation.

$$F0 = \text{sign}(v_{\text{b}})\left(\mu_d + \frac{\mu_s - \mu_d}{1 + \frac{|\Delta v|}{v_0 N(t)}}\right)F_{\text{b}} \quad (4)$$

We can also express this force using variables C_{01}, C_{02}, v_0^h, and y_1^h, but their definitions are too complicated to give here; see [6]. Equations (4) and (5) are solved together; if no real solution exists, or if $v_{\text{b}}\Delta v < 0$, then we reject the solution and set the bow to be sticking ($\Delta v = 0$).

$$F0 = C_{01}(\Delta v + v_{\text{b}} - v_0^h) + C_{02}y_1^h \quad (5)$$

The finger force $F1$ is modelled as an infinitely stiff spring at x_1. Once $a_n(t)$ has been calculated, the bridge signal is:

$$F_{\text{bridge}}(t) = \sqrt{\frac{2}{L}}\sum_{n=1}^{N} a_n(t)\left(\frac{Tn\pi}{L} + EI(\frac{Tn\pi}{L})^3\right) \quad (6)$$

These calculations are performed once for every string, then the bridge signals are added together. The result of the sum is convolved with the impulse response of tapping a violin bridge; the output is our final audio signal.

$y(x,t)$	String displacement at position x, time t
$\phi_n(x)$	Eigenvectors; $\phi_n(x) = \sqrt{\frac{2}{L}}\sin\frac{n\pi x}{L}$
$a_n(t)$	Modal displacement
x_0	Bow position (as a ratio of string length)
x_1	Finger position (ratio of string length)
v_{b}	Bow velocity (m/s)
F_{b}	Bow force (N)
s	String number (selects which string to use)

Table 1. Main variables used in bowed-string algorithm.

4. VIOLINIST'S CONTROL

The virtual violinist must translate musical note data into physical actions. Due to the design goals outlined in Section 1.2, we cannot use theoretical equations (such as Schelleng's "area of stability" [13] or Guettler diagrams [14]) which rely on physical constants. We also avoid "hand-tweaking" any values – the goal is to create a virtual violinist which can perform on any instrument (be it a physical model or a violin-playing robot), so the algorithm should be as general as possible.

Another difficulty is the non-deterministic element $N(t)$ in the physical model – a series of physical actions may produce acceptable or unacceptable output depending on the random seed. To address this problem, a feedback control loop is used (Figure 4).

The violin is a non-linear system; the effects of bowing parameters are not easy to state. A rough generalization is that the bow-bridge distance and velocity together set the overall amplitude, while the bow force determines the timbre and whether Helmholtz motion is achieved [15].

4.1 Pedagogical inspiration and physical parameters

The string s and finger position x_1 can be calculated directly from each note, imitating the "fingerboard tape" often used with students. Although skilled violinists choose different fingerings for expressive purposes, this is not expected from beginning students.

The "bow lanes" allow us to specify the bow-bridge distance x_0 according to the dynamic. The length of bow and note duration fixes the average bow speed. This still allows for variation in the bow speed, but in general beginning violinists do not concern themselves with this. For simplicity we have assumed the bow will accelerate to the target velocity at the beginning of the note, and decelerate to 0 m/s at the end of the note unless it is part of a slur. Parameters x_0 and v_b are then specified from the dynamic (Table 2). More details about these parameters are given in Section 6.

The missing parameter from our description of beginning violin students is the bow force F_b. Our virtual violinist therefore concentrates on bow force, and uses human input during the training to improve the bow force calculations.

4.2 Bow force

Section 5.2 discusses how we calculate our initial force F_b and force modifier K during training. In the control loop, we alter the current F_b according to the selected SVM classifier, which examines the audio output of the physical model. Our audio analysis and machine learning is done with Marsyas [22]. In addition to a typical set of audio features (zero crossings, spectral centroid, rolloff, flux, crest factor, and flatness measure), we also calculate the difference between expected pitch (from s and x_1) and the detected pitch (from the YIN algorithm [23]). We used a window size of 1024, and a hop size of 256. The machine learning produces a class $c \in \{1, 2, 3, 4, 5\}$ (Table 3), which is used to adjust the bow force: $F_\mathrm{b} \leftarrow F_\mathrm{b}\, K^{(3-c)}$.

Dynamic	Bow position x_0 (fraction of L)	Bow velocity v_b (m/s)
f	0.08	0.4
mf	0.10	0.33
mp	0.12	0.26
p	0.14	0.2

Table 2. Physical parameters determined by dynamic.

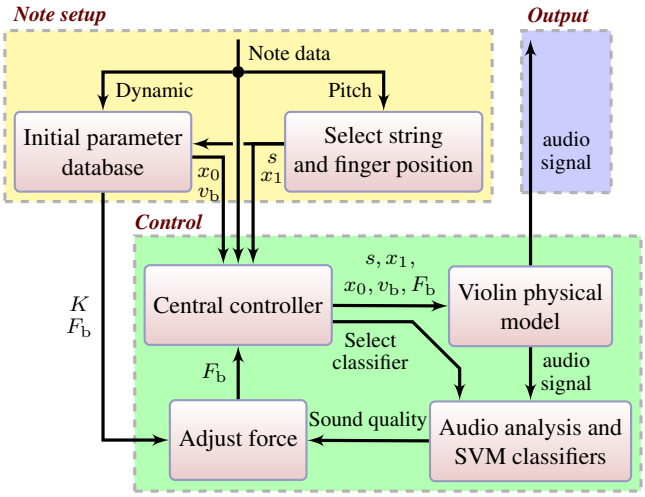

Figure 4. Control details of *Vivi*.

4.3 Central controller

The central controller selects which SVM classifier to use; we train a different classifier for each of the sixteen string-dynamic combinations. The controller also handles the acceleration of the bow, and adds a small amount of Gaussian randomization to F_b and v_b. This imitates the unsteady muscle control of a beginning violinist.

A different SVM classifier was used for each string-dynamic combination because each pair produces a different timbre. The 10-fold cross-validation accuracy supports our intuition: after four hours of training, the individual string-dynamic classifiers were 96–99% accurate, while a single classifier for all G string dynamics was 91% accurate, and a single classifier for all strings played *mf* was 94% accurate.

5. TRAINING

We use human input to help train *Vivi*, but only at the level of a "Suzuki parent" – the human gives feedback about audio (classifying audio as in Table 3), but the precise determination of parameters is done automatically.

Class c	Human judgement of sound
1	not audible
2	"wispy" or "whistling"
3	acceptable
4	"harsh" or "detuned"
5	not recognizable as coming from a violin

Table 3. Classification of audio files.

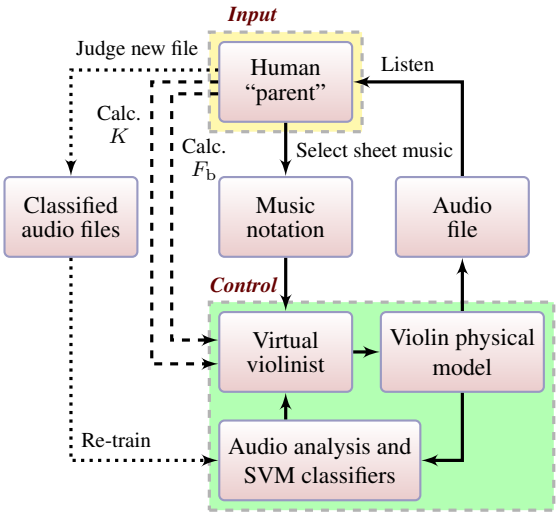

Figure 5. Post-basic training with *Vivi*. The dotted line does not involve the control cycle; the dashed lines involve the control cycle but not the audio output.

5.1 Basic training (human)

Training begins by asking the human to classify audio with a constant bow force. Each audio file is created by running the physical model for 1.0 seconds (to let the note "settle"), then recording the next 0.5 seconds. We begin with $F_\text{b} = 1.0$, then keep on doubling F_b after each audio file is categorized until the user has ranked a file as being category 5. We then return to $F_\text{b} = 0.5$ and keep on halving F_b until the user has ranked a file as being category 1. If any categories were omitted, then we keep on generating new audio files between existing category boundaries until we have at least one file for each category.

We repeated this process for three notes (seen in Figure 6) for each string / dynamic combination. This produced 15–20 audio files for each of the 16 SVM classifiers, with very high accuracy (98–99% for dynamics on the lower three strings, with the E string dynamics' accuracy dropped to 96–97%) with 10-fold cross-validation.

We found that "Anomalous Low Frequencies" (ALF, sometimes referred to as "subharmonics" [24]) were produced quite often on the E string; in one case, we thought that an ALF note was a nicely-played note on the D string! For this reason, we displayed the playing string to the human so that they could correctly identify an ALF note as having too much bow force.

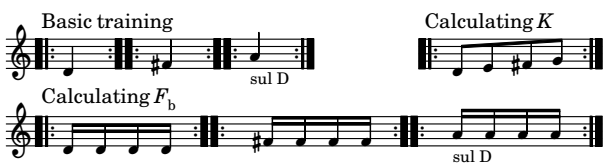

Figure 6. Musical patterns used in training the D string. Repeat signs indicate a variable number of repetitions, depending on human input or the algorithm.

5.2 Determining bow force K and F_b (automatic)

Figure 5 shows an overview of the training process after basic training. The missing parameters for our feedback control (Figure 4) are K and F_b, which we determine automatically from our classified audio files.

5.2.1 Calculating K (one per string-dynamic)

Recall that K determines how much F_b is adjusted according to the sound quality judgement. If K is too small, it will take a long time to reach a good bow force. If K is too large, F_b will oscillate between too much and not enough force. We begin by estimating three initial F_b values based on the maximum force used in the "basic training", F_max. We set $F_\text{b} = \{\frac{1}{16}F_\text{max}, \frac{1}{4}F_\text{max}, F_\text{max}\}$. For example, a D string played *mf* gives $F_\text{b} = \{0.25N, 1.0N, 4.0N\}$.

We define a note's stability cost as (7), in terms of the list C which is all the class values c produced by the trained SVM classifier. C is then split into sub-lists A_i, in which a sub-list ("area") is defined by c changing from below 3 to above 3 (or vice-versa). For example, the list $C = [2, 3, 2, 4, 5, 3, 2]$ would become $A = [[2, 3, 2], [4, 5, 3], [2]]$.

$$\text{Note cost} = \prod_{i}^{|A|} \sum_{c \in A_i} (3 - c)^2 \tag{7}$$

The overall logic is that extreme sounds (categories 1 and 5) are worse than slightly bad sounds (categories 2 and 4). The ideal system would begin with a force below (or above) the correct amount, then quickly move to a good force (category 3). Sounds which alternate between categories 1,2 and 4,5 will maximize the overall multiplication.

We then perform the pattern in Figure 6, once for each initial F_b, and multiply all the note costs together. The resulting number has a large amount of variation, so we repeat this whole process 12 times and take the inter-quartile geometric mean; this is the final cost of a candidate K (Figure 7). We select the candidate K with the lowest cost.

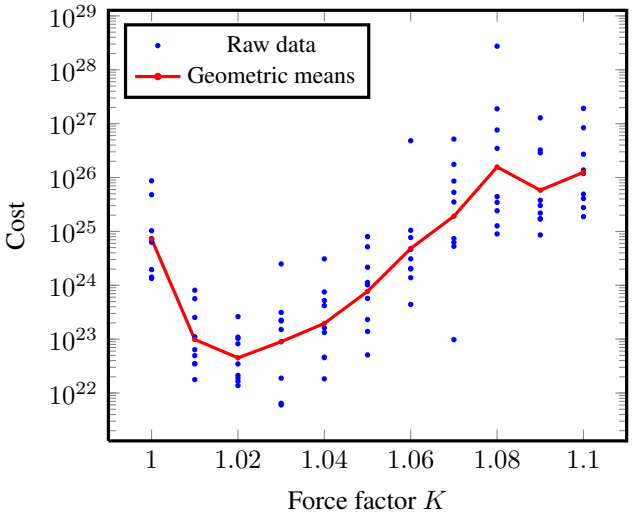

Figure 7. Sample force factors of the D string played *mf*. Extra points were added to show the spread of randomness.

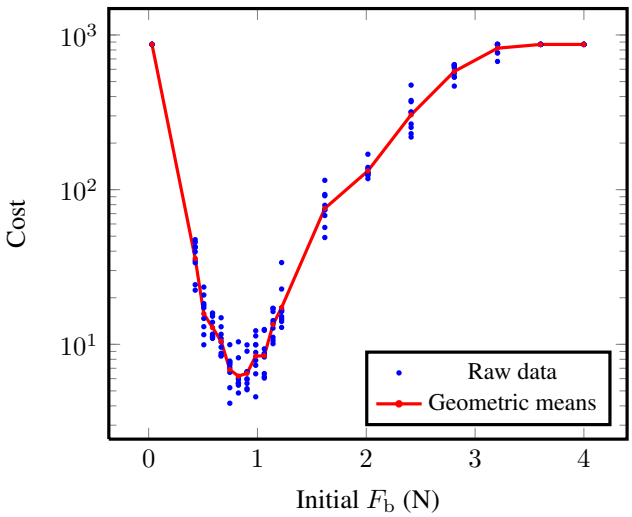

Figure 8. Sample initial forces of the D string played *mf* with an open string, including the "zoomed-in" area. Extra points were added to show the spread of randomness.

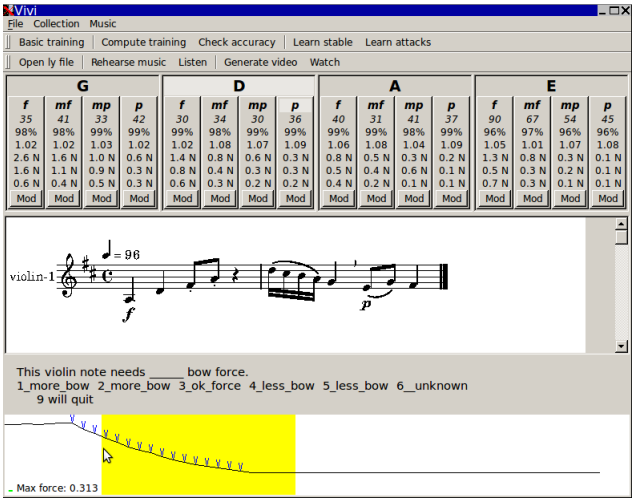

Figure 9. Screenshot of *Vivi*. From top to bottom the numbers represent: number of audio files classified, 10-fold cross-validation accuracy, force factor K, and initial F_b for a finger at 0 (open string), 4, and 7 semitones.

5.2.2 *Calculating initial F_b (three per string-dynamic)*

After calculating K, the real (i.e. not estimated) initial F_b values are calculated from the range F_{min} to F_{max}. As with the calculation for K, we calculate a candidate's cost by playing a musical pattern (1 bar) from Figure 6. The F_b cost (8) is straightforward, but in this case C only represents the classifier values c that were part of the "attack" portion of the note. The attack is over when (9) is true, where L_N is a list of the past $N = 9$ values, and $M = 0.5$. This corresponds to "when the note is stable".

$$\text{Note cost} = \sum_{c \in C} (3 - c)^2 \tag{8}$$

$$M > \frac{1}{N} \sum_{c \in L_N} (3 - c)^2 \tag{9}$$

The individual note costs are multiplied together, then the process is repeated 4 times, and the inter-quartile geometric mean is the overall cost of the candidate F_b. After finding the lowest cost, we "zoom in" to that area by setting F_{min} to the F_b immediately lower than our lowest value, and F_{max} to the F_b higher. The process is repeated, gathering more candidates (Figure 8). We select the candidate F_b with the lowest cost.

This whole process is repeated three times, varying the left-hand finger positions as seen in Figure 6. For the sample D string *mf*, the initial F_b drops from 0.8 N (for an open string) to 0.4 N (for a finger 4 semitones higher) to 0.3 N (finger 7 semitones above open string).

Violinists will notice that this generates an "on the string" bow-stroke. Skilled violinists will vary the amount of initial bow-force considerably, including starting with $F_b = 0$ for an "off the string" bow-stroke. However, the first bow-stroke taught by the Suzuki method is "on the string" – in fact, this bow-stroke is a large contributor to the "Suzuki sound" for which their students are so famous!

5.3 Practicing sheet music (human)

After the human has completed the one-time "basic training" and the computer has completed the automatic determination of K and F_b, the bow control is extremely poor [4].

To improve the bow control, *Vivi* should "rehearse" sheet music selected by the human, who can then listen to the entire performance, or listen to individual notes by clicking on them. For each note, the user can correct any judgements by selecting a portion of a note and indicating what the category should be; this is shown in Figure 9. After correcting a few notes, the user should tell *Vivi* to re-train the SVM classifier, and then re-calculate K and/or F_b. Any sheet music can be used for this interactive practice, but scales are quite useful as they provide few distractions.

6. MUSIC PERFORMANCE

We now turn to the calculation of the note data which is the input to Figure 4. Rather than providing a list of musical notation, we refer to our "black-box" testing framework in Figure 2, which includes all notation understood by *Vivi*.

6.1 Mapping basic notation to note data

Many aspects of musical notation can be translated directly into physical actions with no debate. In the absence of special string markers ("II..."), the written pitch will determine the string and left-hand finger position. A notated slur means that we should not change bow directions; without a slur, we decelerate the bow speed to 0 m/s at the end of the note, and set the next note's direction to be the opposite of the current note.

The markers "pizz." and "arco" are similarly clear – notes that should be pizzicato are simulated as having a very large F_b, pulling the string until the "bow" slips, and then setting F_b to be 0. Arco notes return to the normal bowing.

[4] `http://percival-music.ca/smc2011.html`

Notation	Musical action	Value
Staccato	Shorten duration	0.7
Portato	Shorten duration	0.9
Accent	Increase force	2.5
Last note on a string	Lighten bow force	0.1 s
〃	〃	$0.5F_{\mathrm{b}}$
Breath mark	Shorten duration	0.6
〃	〃	$0.3F_{\mathrm{b}}$
(all)	Max. bow accel.	5.0 m/s

Table 4. *Vivi*'s interpretation of style-specific notation. Unless units are given, all values are expressed as a factor of the unmodified value which would be used.

Text such as "frog", "lh", "mb", "uh", "tip" refers to the position of contact along the bow; these are specified as 0.1, 0.25, 0.5, 0.75, and 0.9. An upbow or downbow mark will set the bowing direction as specified.

6.2 Mapping style-specific notation to note data

Other aspects of musical notation are subject to interpretation or musical styles. For now, we simply hard-coded values for stylistic interpretation, as our focus was on control and training rather than expressive music performance. We listened to *Vivi's* performances of book 1 Suzuki music [4] and adjusted parameters to match the way we expected the pieces to sound. Values are given in Table 4.

(De)crescendi are interpreted as linear interpolation between the beginning and ending dynamics. The physical parameters of fractional dynamics are similarly interpreted as a linear interpolation of the two closest dynamics.

6.3 Variable skill

The timing and intonation can be deliberately degraded to simulate a beginning violinist by applying Gaussian randomization with standard deviation σ to the note's pitch and onset time. If a note occurs on an open string, then we apply no randomization to the pitch, but still alter the timing. We defined 4 skill levels, expressed in pairs of $(\sigma_{\mathrm{pitch}}, \sigma_{\mathrm{timing}})$ from worst to best: (0.5, 0.01), (0.1, 0.005), (0.05, 0.001), (0, 0). Values were estimated by listening.

7. IMPLEMENTATION

Vivi was written primarily in python using the pyqt4 [5] bindings, and is freely available [6] under the GPLv3. Audio analysis and machine learning was performed with Marsyas, while the "training" functionality was written in C++ and made available to the main application with SWIG.

Sheet music was written in LilyPond [7]. In addition to creating PDFs, we extracted musical events from the score by writing event listeners in scheme and attaching them to the relevant portions of the sheet music engraving process.

[5] `http://www.riverbankcomputing.com/static/Docs/PyQt4/html/`

[6] `http://percival-music.ca/vivi.html`

[7] `http://lilypond.org`

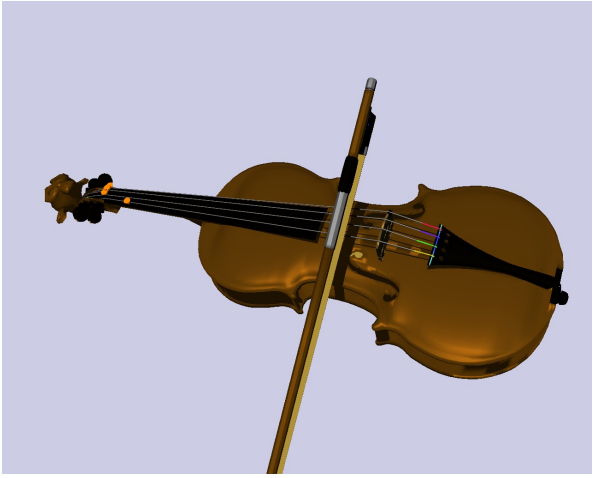

Figure 10. Frame of automatically-produced video. Many thanks to Marcos Press for creating the violin and bow models in Blender, and releasing it under the GPLv3.

The audio analysis and machine learning provides a significant performance penalty compared to running the physical model by itself. Generating the 29 seconds of audio for the music in Figure 2 with the physical model alone took [8] 4 seconds, while the complete feedback control took 17 seconds. Calculating K and the initial F_{b} forces for each string-dynamic was a more substantial bottleneck, taking 326 seconds. Fortunately, this task is embarrassingly parallel, so on a processor capable of running four threads at once it takes approximately 25 minutes to complete all automatic training. In addition to creating an audio file, we also record all physical actions to a file. This can be used to generate a video with Blender [9] (Figure 10).

8. CONCLUSIONS AND FUTURE WORK

We have presented *Vivi, the virtual violinist*; a computer program which can perform sheet music with approximately the skill of a violinist with one year of experience. Intelligent control of the violin physical model is performed with SVM classifiers, and some physical parameters were pre-computed by simulating the performance of the control loop before performing sheet music.

Some major facets of violin music are lacking – we do not perform any glissandi, chords, or vibrato. This is no accident; these skills are not taught to beginning Suzuki violin students. We plan to follow the path of human students, adding features in the same order in which human students learn them. On a similar note, so far we have used the same "style" for all sheet music. This worked for the mostly-Baroque repertoire of book 1, but it will not suffice for later volumes of Suzuki repertoire. We will therefore add the ability for the user to select different "styles", or even automatically select a style based on the composer.

One possibility to improve the control loop is to add haptic feedback to the machine learning. Human violinists using haptic feedback with a violin physical model can exert

[8] Measured on an Intel Core2 Quad CPU running Ubuntu 10.04.

[9] `http://www.blender.org/`

much better control [25]. However, relying on haptic feedback may complicate any future research in using *Vivi* with a robot violinist. Another possibility to improve the sound quality is to perform more training, and/or investigate the use of other machine learning algorithms.

Finally, we would like to teach *Vivi* how to play viola and cello. The physical model should be able to simulate other bowed string instruments, and our supervised training should be easily applicable to other instruments. Our ultimate goal is to allow composers to generate audio for an entire string quartet, given only the sheet music.

9. REFERENCES

[1] J. O. Smith, "Physical Modeling Synthesis Update," *Computer Music Journal*, vol. 20, no. 2, pp. 44–57, 1996.

[2] H. Kenmochi and H. Ohshita, "VOCALOID – commercial singing synthesizer based on sample concatenation," in *Interspeech*, 2007.

[3] M. Hamasaki, H. Takeda, and T. Nishimura, "Network analysis of massively collaborative creation of multimedia contents: case study of hatsune miku videos on nico nico douga," in *UXTV '08*. New York, NY, USA: ACM, 2008, pp. 165–168.

[4] S. Suzuki, *Violin Part Volume 1*. Summy-Birchard Music, 1978.

[5] C. D. Collins, "Connecting Science and the Musical Arts in Teaching Tone Quality: Integrating Helmholtz Motion and Master Violin Teachers' Pedagogies," Ph.D. dissertation, George Mason University, 2009.

[6] M. Demoucron, "On the control of virtual violins: Physical modelling and control of bowed string instruments," Ph.D. dissertation, IRCAM, Paris, 2008.

[7] S. Serafin, "The sound of friction: real-time models, playability and musical applications," Ph.D. dissertation, CCRMA, Standford University, 2004.

[8] E. Bavu, J. Smith, and J. Wolfe, "Torsional waves in a bowed string," *Acta Acustica – Acustica*, vol. 91, N2, p. 241, 2005.

[9] J. Woodhouse and P. Galluzzo, "The Bowed String As We Know It Today," *Acta Acustica – Acustica*, vol. 90, pp. 579–589, July/August 2004.

[10] A. Pérez, "Enhancing Spectral Synthesis Techniques with Performance Gestures using the Violin as a Case Study," Ph.D. dissertation, Universitat Pompeu Fabra, 2009.

[11] E. Maestre, "Modeling instrumental gestures: an analysis/synthesis framework for violin bowing," Ph.D. dissertation, Universitat Pompeu Fabra, 2009.

[12] E. Maestre, A. Pérez, and R. Ramírez, "Gesture sampling for instrumental sound synthesis: violin bowing as a case study," in *International Computer Music Conference*, 2010.

[13] J. C. Schelleng, "The bowed string and the player," *The Journal of the Acoustical Society of America*, vol. 53, no. 1, pp. 26–41, 1973.

[14] A. A. K. Guettler, "Acceptance limits for the duration of pre-Helmholtz transients in bowed string attacks," *The Journal of the Acoustical Society of America*, vol. 101, pp. 2903–2913, 1997.

[15] E. Schoonderwaldt, "Mechanics and acoustics of violin bowing: Freedom, constraints and control in performance," Ph.D. dissertation, KTH, Sweden, 2009.

[16] E. Berdahl, G. Niemeyer, and J. O. Smith, "Feedback Control of Acoustic Musical Instruments," *Technical Report STAN-M-120*, no. 120, June 2008.

[17] E. Berdahl and J. O. Smith, "Inducing Unusual Dynamics in Acoustic Musical Instruments," in *International Conference on Control Applications*, 2007, pp. 1336–1341.

[18] J. Solis, K. Taniguchi, T. Ninomiya, K. Petersen, T. Yamamoto, and A. Takanishi, "Improved musical performance control of WF-4RIV: Implementation of an expressive music generator and an automated sound quality detection," in *Robot and Human Interactive Communication*, Aug. 2008, pp. 334–339.

[19] K. Shibuya, S. Matsuda, and A. Takahara, "Toward Developing a Violin Playing Robot – Bowing by Anthropomorphic Robot Arm and Sound Analysis," in *Robot and Human interactive Communication*, 2007, pp. 763–768.

[20] P. Wrzeciono and K. Marasek, "Violin Sound Quality: Expert Judgements and Objective Measurements," in *Advances in Music Information Retrieval*, Z. Ras and A. Wieczorkowska, Eds. Springer Berlin / Heidelberg, 2010, vol. 274, pp. 237–260.

[21] J. Charles, "Playing Technique and Violin Timbre: Detecting Bad Playing," Ph.D. dissertation, Dublin Institute of Technology, 2010.

[22] G. Tzanetakis, "Marsyas: a case study in implementing Music Information Retrieval Systems," in *Intelligent Music Information Systems: Tools and Methodologies*, S. Shen and L. Cui, Eds. Information Science Reference, 2007.

[23] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[24] M. Kimura, "How to produce subharmonics on the violin," *Journal of New Music Research*, vol. 28, pp. 178–184, 1999.

[25] A. Luciani, J.-L. Florens, D. Couroussé, and J. Castet, "Ergotic Sounds: A New Way to Improve Playability, Believability and Presence of Virtual Musical Instruments," *Journal of New Music Research*, vol. 38, no. 3, pp. 309–323, 2009.